DeepSafety Report – Validation and Improvement of a traffic sign recogntion (TSR)

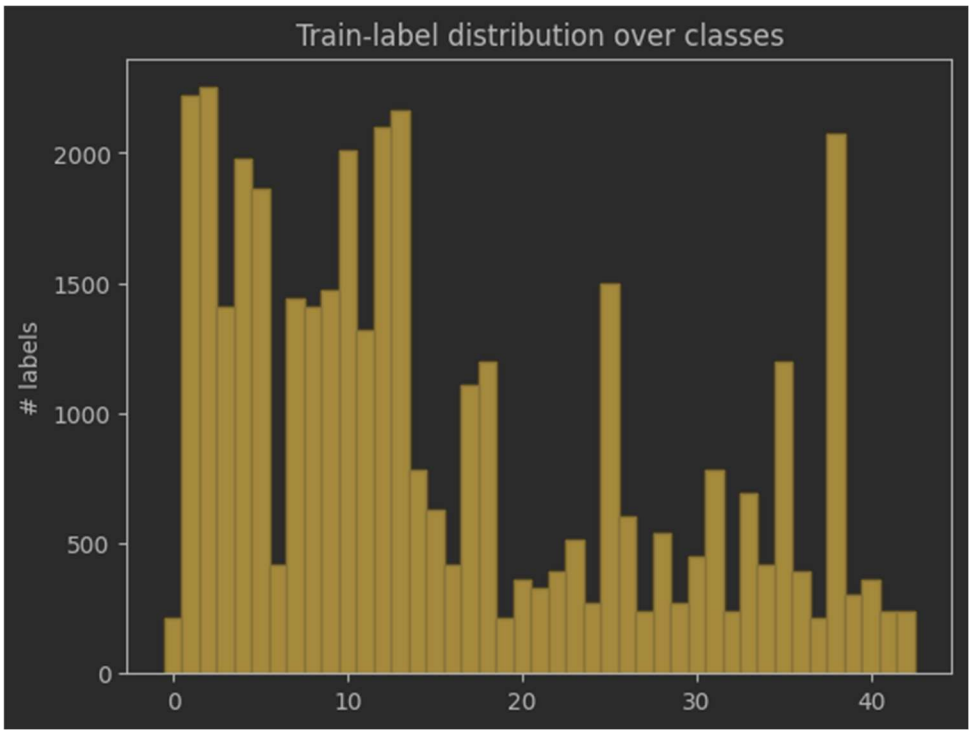
Introduction

The main goal of this report is to implement and validate a working TSR based on the GTSRB traffic sign dataset and the inception feature extractor [1] as base network. While it is relatively simple to train the feature extractor on a given dataset, the validation and performance evaluation on new data, especially real world scenarios is not as easy as it seems: To ensure all safety aspects of a detection system based on deep learning, a variety of cosiderations must be made -including the significance of the train set, reflection of the training procedure, comprehensive validation with a real world representing validation batch and edge case relativation. As a round-off in a secod training iteration the derived improvements should be validated.

First training iteration

The train dataset

The feature extractor is trained on the GTSRB train (sub)dataset which contains 39.000 images of the 42 most important german traffic signs. In a first step the representation of each class is analysed by plotting the quantity distribution over all available classes:

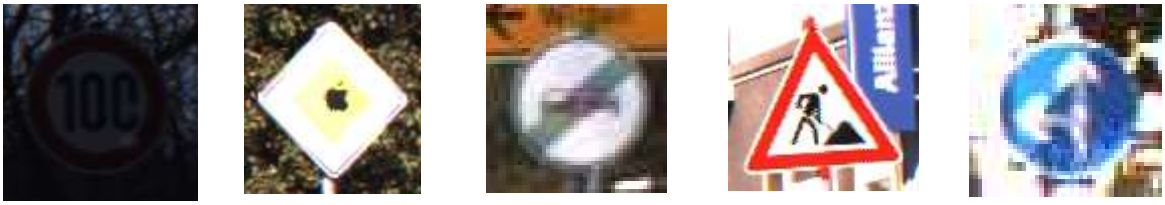


As shown above, the dataset is significantly biased and does not represent all classes with the same quantity. While that may correspond to the occurence of each sign in the real world, a TSR musst detect all possible classes with a sufficient certainty – especially the underrepresented.

By taking a deeper look into the dataset it gets clear that every 30 images are showing one sign, recorded in different sizes and angles, while the collecting vehicle is approaching it. That may have both, advantages and disadvanteges:

- Reduces data set to 1.300 recorded sings, which increases the data bias (i.e. for malformed signs)
- Some signs are captured really rare, with less than 5 signs
- The same signs are shown from different angles and in different sizes, which leads to higher pose variety
- Adaption to real world scenatios: As the vehicle approaches a sign, in a temporal context single-frame-detection-errors (SFDE) can be neglected, because the sign is still deteced in most frames

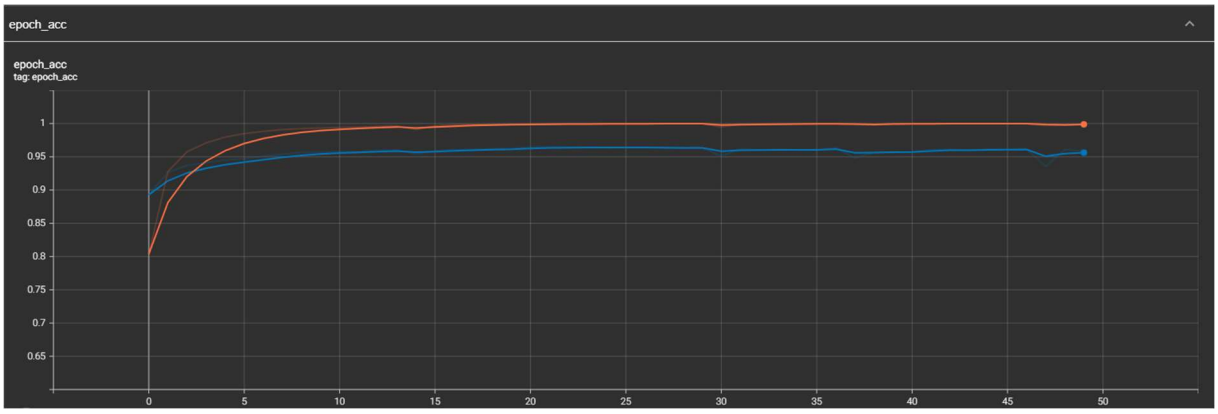
The variety of environmental conditions in which the signs are captured also need to be mentioned: There are really bright or dark images, signs being partial hidden, dirty signs, signs with stickers on it, as well as rotated or tipped ones. Here are some examples:



It can be said that this variety enforces robustness on a big scale dataset, but may also lead to biasing with smaller datasets!

Training results

In a first run the initial inception model is trained for 50 epochs on the original train dataset.



A accuracy of 0.9585 was achived, but might be overfitted compared to the validation-accuracy (0.95). This needs to be evaluated in validation.

Validation

The process of validation was assisted by the metrics average precision, precision and recall for each class and the accuracy and mean average precision (MAP) over all classes. Furthermore thru the use

of a plot of wrong detects and the error matrix as tools, the results could be validated even better. The validation pipeline was validated by five given safetyBatches, the test data set and a new, bigger safetyBatch containing augmented test data, meta data and new images.

Metrics

To validate the results of the trained model, two metrics, the metrics accuracy and map are introduced. These are definde by:

$$MAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

Where:

MAP represents the Mean Average Precision.

N is the total number of classes.

AP_i is the Average Precision for the i-th class.

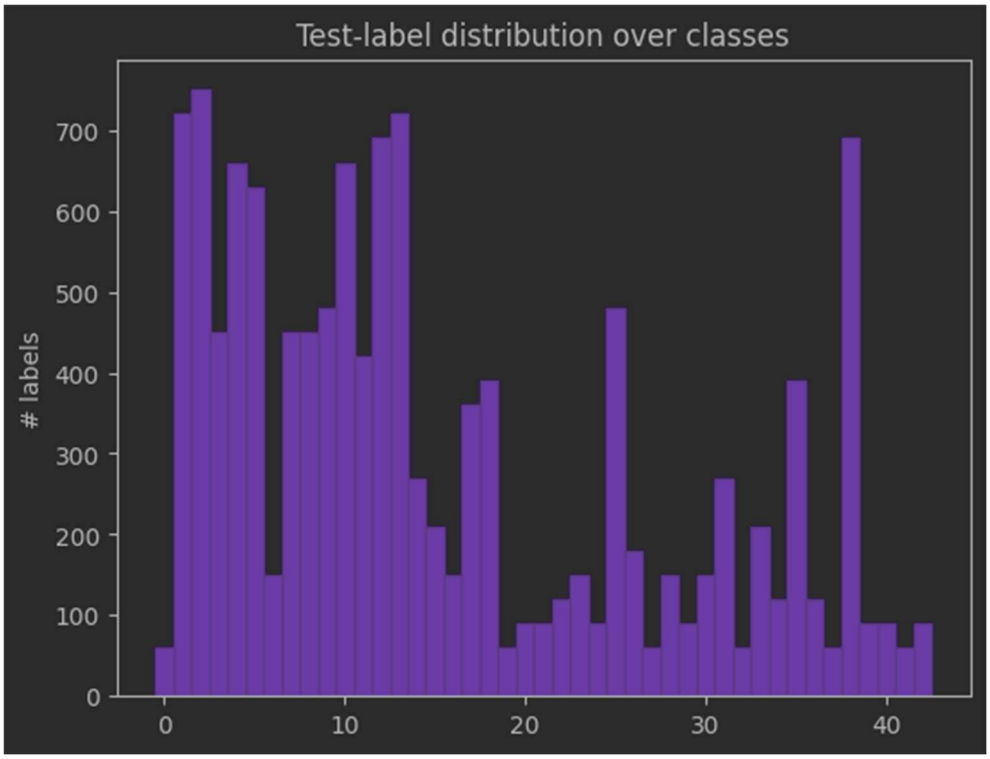
And:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

While accuracy gives a good hint on the ability of the detector to predict the correct class without making errors , the MAP metric indicates how precise the detector predicts each class, considering ist recall.

The test dataset

For validation a second subdataset, the test dataset, provided by GTSRB, is used. It contains 12.500 unsorted images (or 420 signs) with corresponding lables and is the biggest give source of new images.



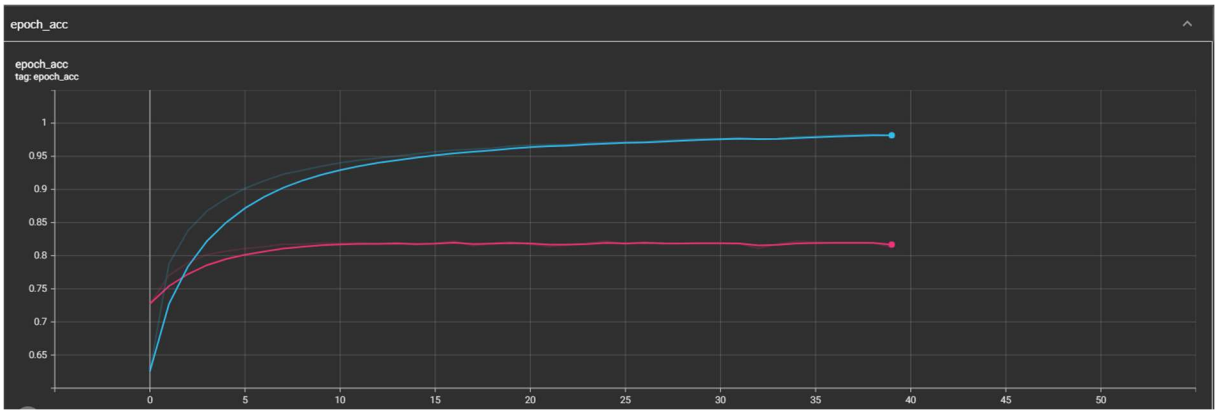
In prior lectures five safety batches were evaluated on their significance:

SafetyBatch	Size (# images)	Contained classes	Accuracy	MAP	features	conclusion
0	16	0, 1, 10, 11, 12	0.9411	0.1129	-	Neither all classes nor representative amount of pictures is included
1	50	42	0.0	0.0232	Two sequences of same sign	Good amount of pictures for this class but other classes missing, especally the one with similar colors and shapes (6, 32, 41)
2	2100	12	0.9971	0.0232	-	One class with many edge cases can give the accuracy for this class but other classes still missing
3	39209	all	0.7447	0.6770	Big validation batch	Is the same as the trainset, thats why also MAP is so high.
4	210	0	0.0619	0.0232	Edgecase: Dark 20 Speed limit signs	Not representative, but shows how model performs in this particular edge case
5	733	-	0.0	0.0	Sunflowers to show false positive rate	Also important that model is able to **not** detect some objects.
6 - FinalBatch	12.630 + 131	all	0.6074	0.3933	New and augmented images over every class	A high variance, augmented validationBatch gives good, representative accuracy for the model, but model could have performed better

Based on the made experience above, a new safety batch is designed. The batch sould contain a representative amount of signs, including all classes, a variety of edge cases, augumented data and different environmental conditions. -> See Validation dataset (*Augumentation is done within the validation process and augmented images are visible afterwards in ValidatiopnAug directory!*)

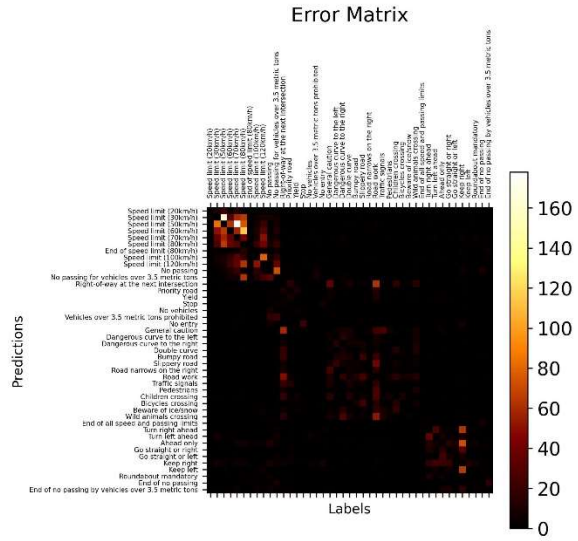
The second iteration

Learning from the first validation an training results, in a second run a more efficient model (EfficientNetB0) was used in combination with a MaxPooling-Layer and augmented training data. Within 39 Epochs a accuracy of 0.9801 could be achieved (validation-accuracy: 0.8186):

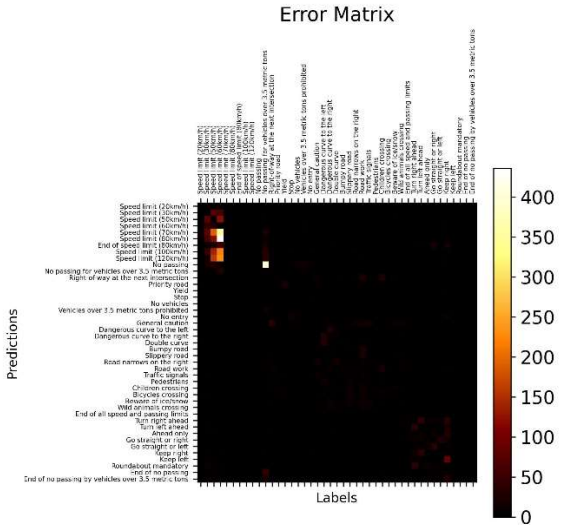


More episodes may give even better results.

The error matrix i. e. looks as follows:



Error matrix after first iteration. Many mismatch with signs of same shape and color. *(for bigger resolution see error_matrix_1.jpg)*



Error matrix after second iteration. While the detector still has problems to detect the different speed limits, rest is doing better. *(for bigger resolution see error_matrix_2.jpg)*

Inspiring work

Yann LeCun & Pierre Sermanet – Traffic Sign Recognition with Multi-Scale Convolutional Networks http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf

Shivank Sharma - GTSRB - CNN (98% Test Accuracy) https://www.kaggle.com/code/shivank856/gtsrb-cnn-98-test-accuracy

Eddie Forson - Recognising Traffic Signs With 98% Accuracy Using Deep Learning https://towardsdatascience.com/recognizing-traffic-signs-with-over-98-accuracy-using-deep-learning-86737aedc2ab